Radiology: Artificial Intelligence

EDITORIAL

Magician's Corner: 9. Performance Metrics for Machine Learning Models

Bradley J. Erickson, MD, PhD • *Felipe Kitamura, MD, MSc*

From the Department of Radiology, Mayo Clinic, 200 First St SW, Rochester, MN 55905 (B.J.E.); and Department of Diagnostic Imaging, Universidade Federal de São Paulo, São Paulo, Brazil and Head of AI, Diagnósticos da América SA, São Paulo, Brazil (F.K.). Received May 27, 2020; revision requested June 16; revision received August 31; accepted October 14. **Address correspondence to** B.J.E. (e-mail: *bje@mayo.edu*).

Authors declared no funding for this work. Conflicts of interest are listed at the end of this article.

Radiology: Artificial Intelligence 2021; 3(3):e200126 • https://doi.org/10.1148/ryai.2021200126 • Content code: AI

"*There are no solutions; there are only trade-offs.*"
Thomas Sowell

In the previous articles, we showed how to process images and train classification models, segmentation models, generative adversarial networks, as well as image denoising models. We also provided a guide on how to visualize model training metrics with TensorBoard and how to connect a model to your picture archiving and communication system. In this article, we will show you how to evaluate the performance of your models by delving into some details about classification and segmentation metrics and show how to calculate them. Performance metrics are useful during model training and validation. We also provide an interactive way for you to see how a custom threshold influences these metrics.

To follow this guide, open Colab by clicking on this link: *https://colab.research.google.com*. Make sure to use Google Chrome as your web browser. You should be prompted to open a file when you start it, but otherwise, select the File > Open Notebook menu option, select the "Github" tab at the top, enter "RSNA" into the search field, and then select the repository "RSNA/Magicians-Corner." Select the entry called "Magicians_Corner_9_Performance_Metrics_for_Machine_Learning.ipynb" to load the notebook. Click the arrow at the left of the first cell to run cell 1 (you may need to accept that it is not Google code).

Classification Metrics

Classification metrics can be divided in three main groups: binary, multiclass, and multilabel. Binary classifiers predict if a sample belongs to one of two possible classes (two-class classifiers). A common use-case for binary classifiers is to answer yes or no questions. An example in radiology would be a neural network that classifies whether a chest radiograph shows pneumonia or not. Multiclass classifiers choose one, and only one, of more than two predefined classes. For example, an image can only be from one of these imaging modalities: MRI, CT, US, or computed radiography. Multilabel classifiers independently answer yes or no for a set of predefined classes. For example, a head CT image with intracranial hemorrhage may have any combination of epidural, subdural, intraventricular, and/or intraparenchymal hemorrhage labels assigned to the image.

Binary Classification

We will start with binary classification metrics, as they serve as a basis to understand the multiclass and multilabel metrics. We will not train a network or do inference in any model in this tutorial, but instead will use results from a classifier applied to a test set, with the inferences stored in the 'scores.csv' file so we can analyze them. You should already have run cell 1 to import the libraries we will use. Then, run cell 2 to download the file scores.csv and cell 3 to show its content (Fig 1). This file contains both the ground truth and the model prediction (inference) for each of 10 200 chest radiographs in a test set. The ground truth was annotated by a radiologist: 0 means normal and 1 means pneumonia. The output probabilities resulting from the inference of this test set in our neural network are decimal numbers in the range between 0 and 1.

As these scores are not binary, we must choose a cutoff value to establish a threshold for the model to make a classification decision of positive or negative. This threshold can be any value between 0 and 1, and the consequences of choosing higher or lower values will be addressed later.

As a first example, let's choose a threshold of 0.5. We will call all cases that had a score greater than the threshold and a ground truth of 1 as true positives. False positives are cases that had a score greater than the threshold, but the ground truth was 0. True negatives are the cases with a score equal to or less than the threshold and a ground truth of 0. False negatives have a score equal to or less than the threshold, but the ground truth was 1. Cell 4 computes the number of cases in each of these categories based on the 10 200 cases in our dataset.

Confusion matrix.— It is common practice to present these four values in a 2 × 2 table, known as the confusion matrix (Fig 2). Run cell 5 to calculate it. The four values in the confusion matrix (true positives, false positives, true negatives, and false negatives) can be used to calculate many other metrics: *(a)* sensitivity (also called the *recall* in the machine learning world), *(b)* specificity, *(c)* false-positive rate, *(d)* false-negative rate, *(e)* positive predictive value (called *precision* in machine learning parlance), *(f)* negative predictive value, *(g)* accuracy, and *(h)* F1 score. Read and execute cells 6 to 13 to understand the formulas for these metrics. The Table summarizes the meaning of each of these metrics and their corresponding synonyms.

	y_pred	y_test
0	0.046000	0
1	0.010214	0
2	0.031231	0
3	0.705250	0
4	0.415613	0
5	0.032076	0
6	0.002251	0
7	0.910970	1
8	0.102211	0
9	0.845478	1

Figure 1: Table of the first 10 cases of our scores.csv file. Each row corresponds to a chest radiograph in our test set. The column y_pred represents the output probability score of our model, while the columns y_test represents the ground truth.

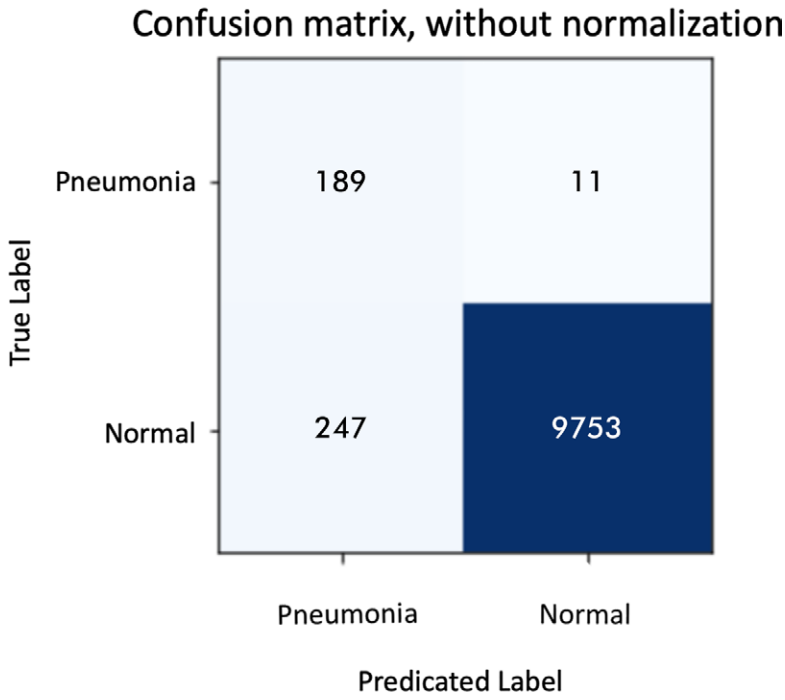

Figure 2: Confusion matrix for the 10 200 cases in our test set with a threshold of 0.5. Rows represent the ground truth and columns represent the predictions.

Metrics as functions of the threshold.— All these metrics are dependent on the threshold we chose (0.5 in our case). Execute cell 14 once and then try different thresholds by sliding the controller and clicking the "calculate" button (note that an error will occur if the cell is not executed before clicking "calculate"). Notice how the number of predicted positives increases as we decrease the threshold. Also, try to find out which threshold is associated with the highest sensitivity and which is associated with the highest specificity.

Cell 15 shows a plot of some metrics as functions of the threshold (Fig 3). Now, run it and check if you were correct in defining the sweet spot for the best sensitivity, specificity, and accuracy. Note that because pneumonia cases are rare, selecting a threshold of 1, where every case is called normal, still results in a high accuracy (though not the highest).

Receiver operating characteristic curve.— Another way to measure model performance is the receiver operating characteristic (ROC) curve (Fig 4). The ROC curve was first used during World War II to study the ability of a radar receiver operator to correctly detect enemy airplanes (versus flocks of birds that were sometimes confused as airplanes). It graphically displays the trade-off between the true-positive rate (sensitivity, or recall) and the false-positive rate (equivalent to 1 minus specificity). It can also be seen as a trade-off between the power and the type I error of the test. Run cell 16 once and adjust the threshold a few times by clicking "calculate." Notice how the threshold is related to a specific point in the ROC curve, but it does not change the curve itself. The closer the ROC curve is to the upper left corner, the better the model is overall. The area under the ROC curve (AUCROC) summarizes how good a test is regardless of the threshold but does not define an operating model—one must select the threshold to actually put the tool into practice. If we randomly draw one positive case and one negative case from our test set, the probability that the score of the positive case will be greater than the score of the negative case is the AUCROC. An AUCROC of 1 is the perfect test, while an AUCROC of 0.5 is the worst-case scenario, equivalent to flipping a coin. Finally, in cases where very high sensitivity or specificity is the goal, it may be that while one model has a higher AUCROC, another may be better for a task demanding high sensitivity or specificity. AUCROC is not dependent on disease prevalence, so it is not a good metric in cases with class imbalance because it may overestimate performance in that setting.

Precision-recall curve.— It is also common to plot the trade-off between precision (positive predictive value) and recall (sensitivity). This is the precision-recall (PR) curve. Run cell 17 to plot the PR curve and try different thresholds again. Note that, as with the ROC curve, the threshold defines a specific point on the PR curve, but it does not change the curve itself. The area under the PR curve (AUCPRC) can be calculated by the average precision score, and it is not influenced by the threshold either. AUCPRC is dependent on disease prevalence because precision is dependent on prevalence, so it is a good choice for a scenario with class imbalance.

Summary of the Meaning of Each Metric Covered in this Guide

Metric	Meaning	Synonyms	Mathematical Formula
Sensitivity	The fraction of positive cases predicted as positive	Recall, true-positive rate	$\text{Sensitivity} = \frac{TP}{TP + FN}$
Specificity	The fraction of negative cases predicted as negative	Selectivity, true-negative rate	$\text{Specificity} = \frac{TN}{TN + FP}$
False-positive rate	The fraction of cases predicted positive that were actually negative	Fall-out, probability of false alarm	$\text{FPR} = \frac{FP}{TN + FP}$
False-negative rate	The fraction of cases predicted negative that were actually positive	Miss rate	$\text{FNR} = \frac{FN}{TP + FN}$
Positive predictive value	The fraction of truly positive cases from all cases the model predicted positive	Precision	$\text{PPV} = \frac{TP}{TP + FP}$
Negative predictive value	The fraction of truly negative cases from all cases the model predicted negative	None	$\text{NPV} = \frac{TN}{TN + FN}$
Accuracy	The fraction of cases the model correctly predicted	None	$\text{Accuracy} = \frac{TP + TN}{TP + FN + TN + FP}$
F1 score	The harmonic mean of positive predictive value and sensitivity	F score, F measure, Dice similarity coefficient	$\text{F1} = \frac{2TP}{2TP + FP + FN}$

Note.—FN = false negative, FNR = false-negative rate, FP = false positive, FPR = false-positive rate, NPV = negative predictive value, PPV = positive predictive value, TN = true negative, TP = true positive.

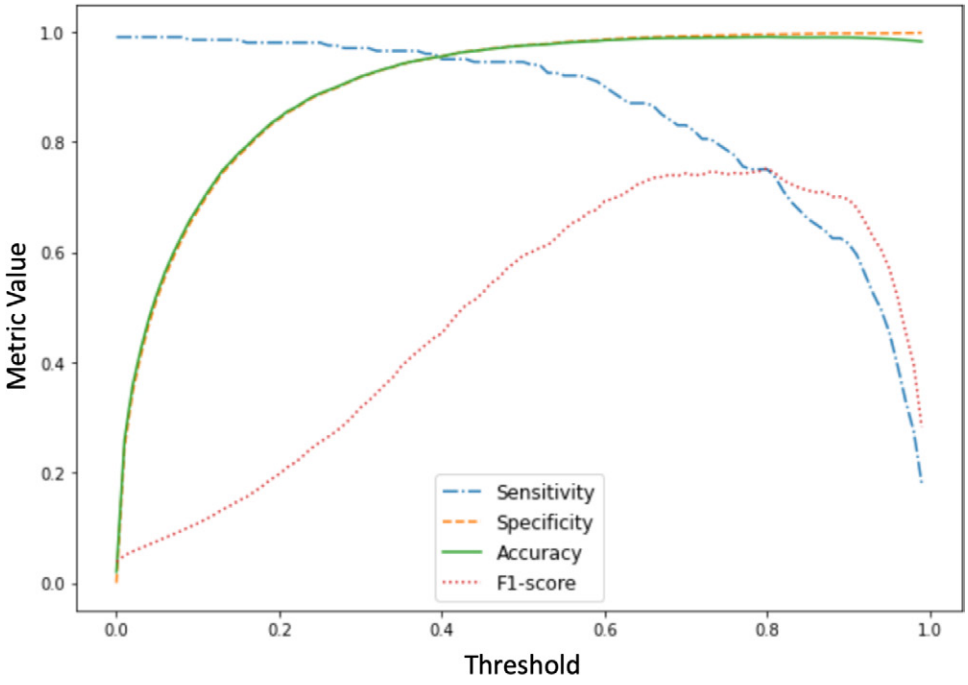

Figure 3: Sensitivity, specificity, accuracy, and F1 score as a function of the threshold.

Metric dependencies on prevalence and threshold.— Accuracy, positive predictive value, negative predictive value, AUCPRC, and F1 score are functions of the prevalence, while AUCROC, sensitivity, specificity, false-positive rate, and false-negative rate are not. The main issue with metrics that are not dependent on prevalence is their incapacity to show the drop in performance in situations with class imbalance. An example is mammography screening where there is very low disease prevalence. One could train a model with high accuracy, sensitivity, specificity, and AUCROC, but very low F1 score, AUCPRC, and positive predictive value. It could be alluring to report only the accuracy, sensitivity, specificity, and AUCROC, while the correct thing to do would be to report all of them, along with the disease prevalence.

Accuracy, positive predictive value, negative predictive value, F1 score, sensitivity, specificity, false-positive rate, and false-negative rate are functions of the threshold, whereas AUCROC and AUCPRC are not.

Multiclass and multilabel.— Most of the binary classification metrics are easily extended to work with multiclass and multilabel classifiers. However, there are multiple ways to calculate each of the binary metrics as multiclass and/or multilabel.

"Micro" counts the total number of true and false positives and negatives and calculates the metrics globally. "Macro" is the simple mean of the metric calculated separately for each label. It is not influenced by prevalence. "Weighted" is similar to macro, but the mean is weighted by each respective class frequency. For "Samples," the metric is calculated in the instance level and then averaged. Run cell 18 to calculate F1-micro and F1-macro in an example of a three-class classifier that predicts normal, pneumothorax, and pneumonia in chest radiographs.

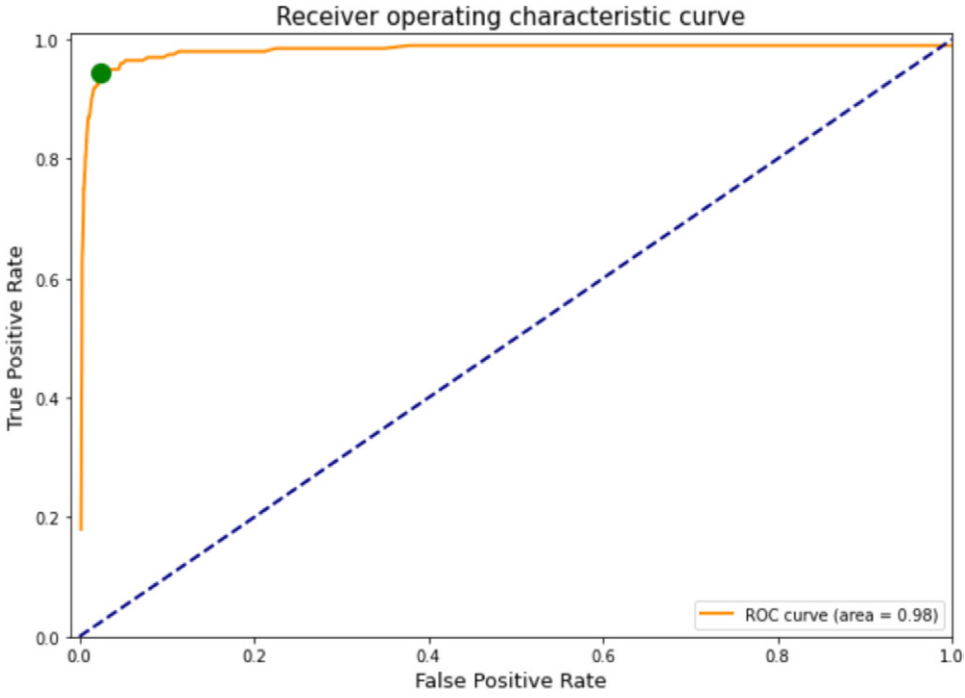

Figure 4: Receiver operating characteristic curve. This curve shows the trade-off between sensitivity and false-positive rate.

When one has multiple classes, the AUCROC can be done in one of two ways: "One-vs-one" computes the AUCROC for all pairwise combinations of classes. "One-vs-rest" computes the AUCROC for each class versus all other classes. Run cell 19 to compute the "one-vs-rest" AUCROC in the three-class classifier problem mentioned above.

Another metric that can be used for both binary and multiclass problems is Cohen kappa score. This metric is usually applied to calculate inter- and intrarater agreement. A kappa of 0 means no agreement and 1 means perfect agreement. To calculate kappa, we need to define if we want linear, quadratic, or no weighting. No weighting means we punish the metric equally for any misclassification. This makes sense in the case of categorical variables, where the categories have no intrinsic ordering (pneumonia, pneumothorax, normal). For ordinal variables (ie, American College of Radiology mammogram breast density categories A, B, C, and D), we may want to punish a class A that was misclassified as D more than a class A that was misclassified as B. In this case, we can use linear or quadratic weights. Run cell 20 to calculate Cohen kappa score in the three-class chest radiograph example mentioned above.

Calibration.— The model output can be considered a probability that a given case is 1 (pneumonia in our case). However, this probability may not be well calibrated, meaning it could be higher or lower than it should. To measure how well the output probabilities are calibrated, we will plot a curve that shows the prevalence of 1 (ground truth positive) as a function of the threshold (Fig 5). Run cell 21 to calculate the calibration plot for our binary pneumonia classifier. A perfectly calibrated model will have a linear correlation (x = y). A calibrated model may be required in some circumstances, but not always. In the case of an artificial intelligence tool that is expected to output only a binary response (pneumonia vs no pneumonia), there is no need to calibrate the model's output. However, if this same tool is expected to show the radiologist a probability of pneumonia, the output needs to be calibrated so it reflects a real probability, because an uncalibrated score can be misleading. Figure 5 shows the output of an uncalibrated model, and from that graph you may observe that for instance, an output of 0.6 (x axis) corresponds to less than 0.2 probability of pneumonia (y axis).

Segmentation Metrics

Classifier metrics measure the ability to make a prediction that is generally applied for an image or set of images. While one can view segmentation as classification at the pixel level (this pixel is or is not a part of the object of interest), there are other metrics that are more frequently applied for segmentation.

Overlap Methods

The most common segmentation metric for medical images is probably the Dice-Sørensen coefficient, also known as the Dice similarity coefficient (both abbreviated as DSC). The definition is that it is two times the intersection of the predicted image and the truth image divided by the union of the truth plus the predicted image. We use two times the intersection so that we have a range from 0 (where the pre-

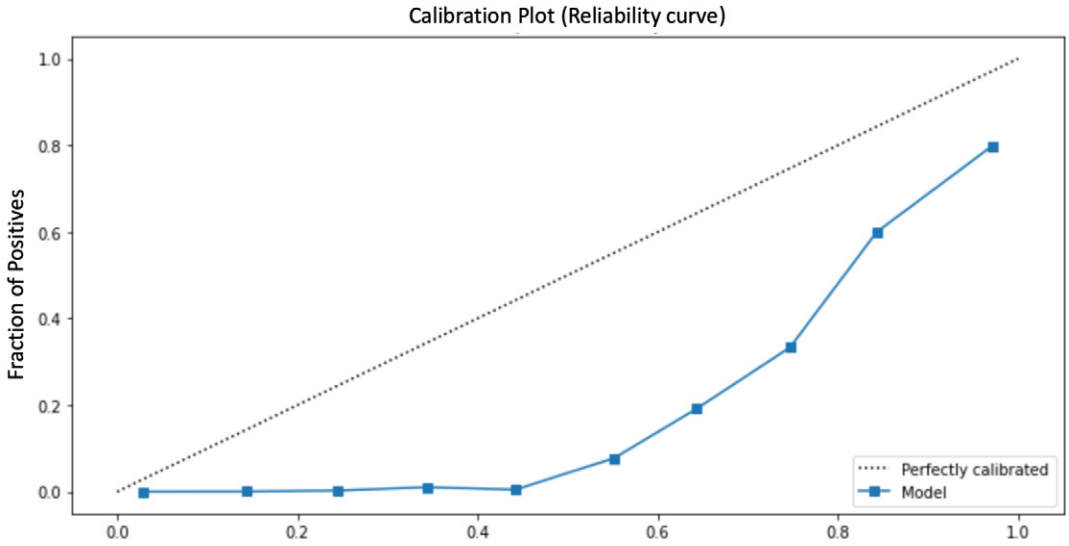

Figure 5: Calibration plot is a curve that shows the prevalence of 1 (ground truth positive) as a function of the threshold. This is an uncalibrated model because the blue curve does not fit the diagonal. In this example, the output of the model might be 0.6, but the probability of the prediction is actually about 0.2.

diction is completely wrong) to 1 (where the prediction is exactly right). Note that in some cases, the truth can be that there are no object pixels in an image, and if the prediction is correct (no pixels set), then the divisor is 0, and attempting to calculate this will result in a divide by 0 error. For that reason, we usually add a small number to the divisor to assure there is no divide by 0 error, or one can check for this condition before dividing and instead return a predefined value. This is sometimes called a soft Dice.

Note that we have usually used a loss function that has a lower value when getting closer to the correct answer, while the DSC has larger values when more correct. Therefore, we typically also create a 'Dice_Loss' function which is usually 1 – DSC. Cell 22 defines the DSC and Dice_loss functions—please run that cell now.

Distance Methods

Another popular segmentation metric is the Hausdorff distance (HD) (1). The HD is the distance between the edge of the "truth" object and the "predicted" object. There are actually several metrics based on HD, including the maximum distance, the average distance, and various percentiles (eg, 80% of the edges are less than some threshold distance). One of the more popular variants is called the modified Hausdorff distance (MHD) (2), which usually more closely matches human rankings of overlap. We note here also that computing HD family metrics tends to be more difficult in three dimensions, as the closest edge may be in a different section. Run cell 23 to define these two functions.

In cell 24, we define a function that will create an image of a defined size and also with a defined "object" within it. We will use this function to create simple phantom images that will serve as both truth and prediction. Note that the image size must be the same for both truth and prediction, but the shape of the "on" part can be different. We will experiment with different "on" shapes to see how it affects the metrics. The cell also defines a function to display the combination of the truth and prediction images so that you can see what is going on. Please run cell 24.

In cell 25, we test the metrics with a digital phantom image. The base image is 100 pixels in X and Y direction, and the "truth" object is a 51 × 51 square in the middle of the image. The first prediction image made is slightly off—it is a rectangle that is slightly wider and shorter than the true object. Run the cell to see the metrics. Now try other settings for the prediction, including when the prediction is exactly right, but some other variations.

Returning to cell 24, note that "nearest" interpolation (interpolation is a mathematical method to add new data points to a discrete set of known data points) is used for the image display function. That is because if some other interpolation function like "linear" was used, the imshow () function would linearly interpolate the image to make it fit the display area. In that process, it would create mask values that are not valid. For instance, in our case, the truth mask value is 2, and prediction is 1; when they are both "on" the value is 3 (true positive). In cases where the truth is "on" but the prediction is wrong, the value is 2 (false negative), false positives are 1, and true negatives are 0. Now consider the case where a true positive is next to a true negative, and the display function has to linearly interpolate, perhaps creating a new value of 1.5, which it might then round to 2. In that case, while the prediction is correct, the displayed image would suggest it is wrong. This is not just a display issue—when one augments images for segmentation (eg, slight rotation), while the image may be linearly interpolated, the mask must use nearest neighbor to maintain the meaning of the mask. Try changing the interpolation parameter to "bilinear" and rerun cells 24 and 25 and see the impact (Fig 6).

In cell 26 we will try some specific types of errors to see the impact. Some medical structures are large and smooth, and thus have a rather low surface-to-volume ratio. These

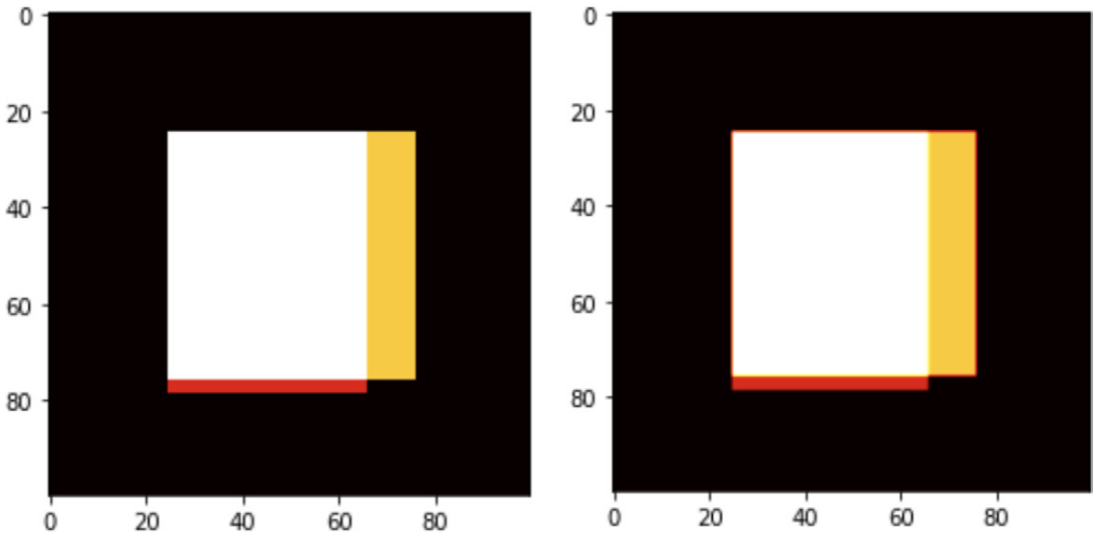

Figure 6: Visual display of image segmentation maps. The left shows the correct map while the one on the right shows a thin rim of red suggesting an error in segmentation. This is an artifact of the display routine if linear interpolation is used, and the image size does not match the display size well. It emphasizes the importance of using nearest neighbor interpolation on mask images when augmentations like rotation or scaling are applied.

tend to have good DSCs even when there are small parts that are way off. Run cell 26 unmodified to confirm that you get the correct display—it should be a white square on a black background.

Now, uncomment (delete the '#') line 8 (with "#Cell #26" being counted as the first line) to test the first variation we will try: The prediction matches the truth square, except for a line that goes another 20 pixels from the right edge. Run cell 26 again to visually confirm this, and observe the effect on DSC, HD, and MHD.

Next, put the comment back on line 8 (add a "#"), uncomment lines 11 and 12, and run cell 26 again. Now we are off by just 1 pixel, but for a large part of the boundary of the rectangle. This causes some reduction in DSC, but minimal impact on HD and MHD (the values we calculated are in the comment to also help you compare).

Finally, put the comments back for lines 11 and 12, and uncomment lines 15 through 18 and run the cell. Here we create a long and narrow truth, and again have only 1 pixel offset for the prediction. Note that the DSC and MHD drop more while the HD is unchanged.

Performance Metrics versus Training Metrics

While the above metrics are very useful for understanding the performance of a deep learning tool (and indeed almost any diagnostic tool), there are other metrics that are often used while training a deep learning algorithm. For instance, binary cross entropy is a very popular metric for measuring a network's performance, and the loss curves that we display (such as in Vogelsang et al [3]) reflect these metrics. There isn't room here to discuss cross-entropy calculations, but a nice explanation is provided in *https://machinelearningmastery.com/cross-entropy-for-machine-learning/*. However, these other metrics often do not have good correlation with aspects of clinical patient care. On the other hand, some of the performance metrics we describe are not differentiable (a differentiable function is one whose derivative exists at each point of its domain), which is a requirement for training. Fortunately, most artificial intelligence training tools allow one to compute and display one or more of the performance metrics we have described, even if they aren't used as the training metric for adjusting the weights during training.

Conclusions

The metrics used in an artificial intelligence project will have a substantial impact on what the artificial intelligence tool actually does. In the case of classification, there are many valid ways to measure how well a classifier does. While the AUCROC does a nice job of characterizing the overall performance, it does not define a specific operating point (that is, the threshold value to be used in practice). In cases where high sensitivity or specificity are more important than accuracy, a tool with a lower AUCROC might actually be the best choice. It is also very helpful to see the confusion matrix when training a classifier to understand the nature of errors being made by your classifier. There are also several segmentation metrics that can be used, and the "best" will depend on the specific task at hand. Cases where the structure has a very irregular shape may be better characterized by using a distance metric like MHD while more regular structures may do best with the DSC. A good understanding of the various performance metrics will likely help you understand why your artificial intelligence tool performed a certain way, and what you might do to improve its performance.

Author contributions: Guarantors of integrity of entire study, B.J.E., F.K.; study concepts/study design or data acquisition or data analysis/interpretation, B.J.E., F.K.; manuscript drafting or manuscript revision for important intellectual content, B.J.E., F.K.; approval of final version of submitted manuscript, B.J.E., F.K.; agrees

to ensure any questions related to the work are appropriately resolved, B.J.E., F.K.; literature research, B.J.E., F.K.; experimental studies, B.J.E., F.K.; statistical analysis, F.K.; and manuscript editing, B.J.E., F.K.

Disclosures of Conflicts of Interest: B.J.E. Activities related to the present article: serves as consultant to the editor for *Radiology: Artificial Intelligence*. Activities not related to the present article: disclosed no relevant relationships. Other relationships: disclosed no relevant relationships. **F.K.** Activities related to the present article: disclosed no relevant relationships. Activities not related to the present article: author is consultant for MD.ai; author is head of AI at DASA. Other relationships: disclosed no relevant relationships.

References

1. Hausdorff F. Grundzüge der Mengenlehre. Leipzig, Germany: Veit, 1914.
2. Dubuisson M, Jain AK. A modified Hausdorff distance for object matching. In: Proceedings of 12th International Conference on Pattern Recognition, Jerusalem, Israel, October 9–13, 1994. Piscataway, NJ: IEEE, 1994.
3. Vogelsang DC, Erickson BJ. Magician's Corner: 6. TensorFlow and TensorBoard. Radiol Artif Intell 2020;2(3):e200012.